

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-334468

(43)Date of publication of application : 22.12.1995

(51)Int.Cl.

G06F 15/16

G06F 15/16

(21)Application number : 06-125484

(71)Applicant : TOSHIBA CORP

(22)Date of filing : 07.06.1994

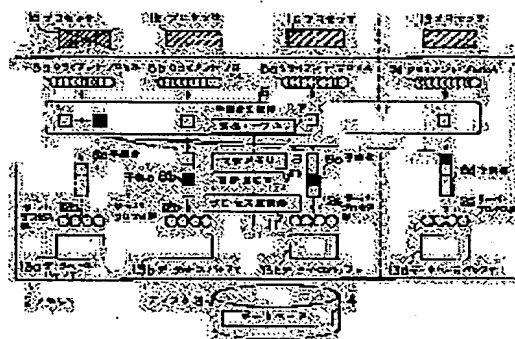
(72)Inventor : KATO NORIHIRO

(54) LOAD DISTRIBUTION SYSTEM

(57)Abstract:

PURPOSE: To provide a load distribution system capable of efficiently distributing load when the rearrangement of processes among respective processors due to a processor fault or the like is executed.

CONSTITUTION: This load distribution system is provided with a process monitoring part 11 for monitoring which processor client processes 5a to 5d and server process groups 12a to 12d are arranged under the control of, a procedure distributing part 6 for distributing a procedure requested from the client processes 5a to 5d to prescribed server process groups 12a to 12d and a control means for controlling the distributing part 6 so that procedures requested from plural client processes 5a, 5b are distributed to one server process group 12b at the time of detecting a processor 1b in which plural client processes 5a, 5b and plural server process groups 12a, 12b are arranged due to the generation of a fault in the other processor 1a.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-334468

(43) 公開日 平成7年(1995)12月22日

(51) Int.Cl.⁶

G 0 6 F 15/16

識別記号

3 8 0 Z

4 7 0 S

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数 2 O L (全 7 頁)

(21) 出願番号 特願平6-125484

(22) 出願日 平成6年(1994)6月7日

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 加藤 宣弘

東京都青梅市末広町2丁目9番地 株式会

社東芝青梅工場内

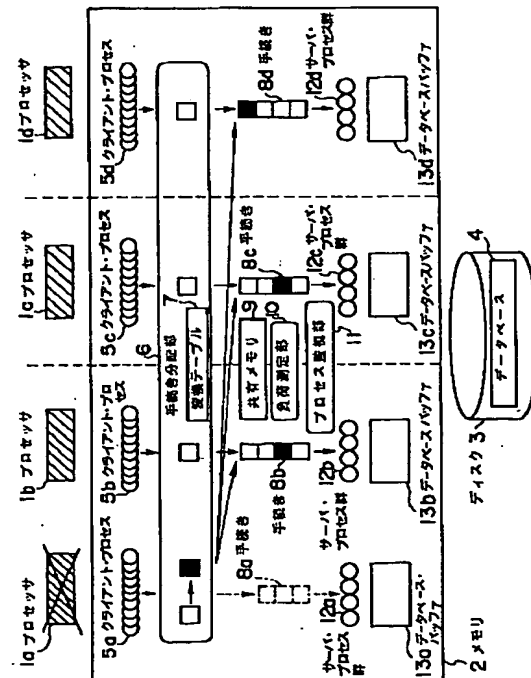
(74) 代理人 弁理士 鈴江 武彦

(54) 【発明の名称】 負荷分散方式

(57) 【要約】

【目的】プロセッサ障害等によるプロセッサ間でのプロセス再配置が行われた際に、効率的に負荷分散を行うことを可能とする負荷分散方式を提供する。

【構成】クライアント・プロセス5a～5d及びサーバ・プロセス群12a～12dがいずれのプロセッサ下に配置されているのかを監視するプロセス監視部1・1と、上記クライアント・プロセス5a～5dから要求された手続きを所定のサーバ・プロセス群12a～12dに分配する手続き分配部6と、他のプロセッサ1aの障害により複数のクライアント・プロセス5a～5b及びサーバ・プロセス群12a～12bが配置されているプロセッサ1bを検出した場合に、この複数のクライアント・プロセス5a～5bから要求された手続きが一つのサーバ・プロセス群12bに分配されるように上記手続き分配部6を制御する制御手段とを具備してなることを特徴とする。



【 特許請求の範囲】

【 請求項1 】 複数のプロセッサ各々の制御下に、クライアント・プロセスと、このクライアント・プロセスから要求された手続きを所定のメモリ領域を共有しながら実行する複数のプロセスからなるサーバ・プロセス群とを配置し、いずれかのプロセッサに障害が発生したときに、そのプロセッサ下に配置されていた上記クライアント・プロセス及びサーバ・プロセス群を他の特定の

10 プロセッサ下に再配置させることにより処理を継続するマルチプロセッサ計算機において、
上記クライアント・プロセス及びサーバ・プロセス群がいずれのプロセッサ下に配置されているのかを監視するプロセス監視手段と、上記クライアント・プロセスから要求された手続きを所定のサーバ・プロセス群に分配する手続き分配手段と、上記プロセス監視手段の監視結果により複数のクライアント・プロセス及びサーバ・プロセス群が配置されているプロセッサを検出した場合に、この複数のクライアント・プロセスから要求された手続きが一つのサーバ・プロセス群に分配されるように上記

20 手続き分配手段を制御する制御手段とを具備してなることを特徴とする負荷分散方式。
【 請求項2 】 各プロセッサの負荷を測定する負荷測定手段と、この負荷測定手段の測定結果によりプロセッサ間に負荷の偏りがあると判断した場合に、負荷の高いプロセッサ下のサーバ・プロセス群に本来分配されるべき手続きの一部を負荷の低い他のプロセッサ下のサーバ・プロセス群へ分配されるように上記手続き分配手段を制御する制御手段とを具備してなることを特徴とする請求項1記載の負荷分散方式。

【 発明の詳細な説明】

【 0001 】

【 産業上の利用分野】本発明は、フォールト・トレランスを実現するマルチプロセッサ計算機に適用して好適な負荷分散方式に係り、特にプロセッサ障害等によるプロセッサ間でのプロセス再配置が行われた際に、効率的な負荷分散を行うことを可能とする負荷分散方式に関する。

【 0002 】

【 従来の技術】従来のフォールト・トレラント計算機の中には、あるプロセッサが障害を起こした場合に、その
40 プロセッサ下で実行していたプロセスを他のプロセッサ下で継続して実行することのできる計算機がある。そのような特徴を持つフォールト・トレラント計算機として、例えば、文献「P.A.Bernstein, "Sequoia: A Fault-Tolerant Tightly Coupled Multiprocessor for Transaction Processing", IEEE COMPUTER, Vol. 21, No. 2, Feb. 1988」で述べられているようなノンライト・スルー・キャッシュを用いたマルチプロセッサ計算機がある。このような計算機においてオンライン・トランザクション処理を行う場合には、データベースにアクセスするサーバ

2
・プロセスを特定のプロセッサで実行するようなプロセス構成を取る。それは以下のような理由による。

【 0003 】ノンライト・スルー・キャッシュを採用するマルチプロセッサ計算機の場合には、キャッシュ内での共有データの書き換えがすぐにメモリへ反映されないために、他のプロセッサによる共有データへのアクセスは、それがメモリへ書き戻されるのを待たなければならない。この待ち時間はプロセッサの使用効率を低下させることとなる。しかし、同じプロセッサ内で動作する他の
10 プロセスが共有データにアクセスする場合には、キャッシュ・ヒットするために、メモリへの書き戻しを待つ必要がない。従って、データベースにアクセスするサーバ・プロセスのように、共有データにアクセスする複数のプロセスはできる限り同じプロセッサで実行するのが望ましいと言える。

【 0004 】一方、オンライン・トランザクション処理を実行する場合には、システムのスループットを最大限引き出すために、プロセスを複数のプロセッサを並列に動作させる必要がある。

20 【 0005 】このような二つの相反する要件を満足するためには、グループ間でのデータの共有度ができる限り低くなるようにプロセスをグループ化し、かつ、それぞれのグループを特定のプロセッサに割り付けるようにする。つまり、同じグループに属するプロセスは同じプロセッサ下で動作するので待ち時間が少なくなり、別のグループに属するプロセス相互間ではデータの共有度が低いので待ち状態になる場合が低くなるため、プロセッサを効率的に使用することができる。

30 【 0006 】従来のオンライン・トランザクション処理システムは、トランザクション・モニタと呼ばれるソフトウェアを用いて、トランザクションをクライアント・サーバのプロセス構成で処理する。図6にトランザクション・モニタを用いたプロセス構成を示す。

【 0007 】クライアント・プロセス5a～5dが実行してほしい手続き8a～8dをコールすると、トランザクション・モニタがその手続き8a～8dを実行可能なサーバ・プロセス群12a～12dのキューに引き渡す。サーバ・プロセス群12a～12dはキューから要求を取り出して指定された手続き8a～8dを実行した後、クライアント・プロセス5a～5dへ応答する。

【 0008 】このようなトランザクション・モニタを上述のフォールト・トレラント計算機で動作させる場合、以下の3つの方針に基づいて構成することにより、高性能なシステムを構築できる。

【 0009 】まず第一に、データベース・バッファを共有するサーバ・プロセスを同じプロセッサに割り付ける。この理由は上述の通りである。第二に、一つのプロセッサに割り付けられた全てのサーバ・プロセスに対して、一つのデータベース・バッファを共有させる。なぜならば、異なるデータベース・バッファで同じデータを

3

共有する場合、データの一貫性を取るための制御が必要であり、その制御はかなりオーバーヘッドの高い処理となるからである。第三に、システムのスループットを最大限に引き出すために、すべてのプロセッサにおいてほぼ均一な負荷でサーバ・プロセスを動作させる。以上の方針に基づいたフォールト・トレラント 計算機におけるランザクション・モニタの構成例を図6 に示している。なお、ここでは、一つのデータベース・バッファを共有する複数のサーバ・プロセスをサーバ・プロセス群と呼ぶこととする。

【 0 0 1 0 】

【 発明が解決しようとする課題】 図6 に示すような構成で、あるプロセッサに障害が起こった場合、フォールト・トレラント 計算機のリカバリ 機能が働き、障害プロセッサで実行していたサーバ・プロセスを自動的に他の特定のプロセッサで実行させることになる。

【 0 0 1 1 】 この結果、一つのプロセッサで複数のサーバ・プロセス群を動作させることになり、また、障害プロセッサの処理を引き継いだプロセッサの負荷が高くなるためにプロセッサ間での負荷が不均等になる。そのため

【 0 0 1 2 】 いま、図6 に示すプロセッサ1 a に障害が発生したとすると、図7 に示すようにクライアントプロセス5 a 及びサーバ・プロセス群1 2 a は、プロセッサ1 b 下で動作することとなる。即ち、プロセッサ1 b では2 つのサーバ・プロセス群1 2 a ～1 2 b が動作することになる。従って、2 つのデータベース・バッファ1 3 a ～1 3 b に重複してデータを持つ可能性があり、データの一貫性を保証するための制御にかかるオーバーヘッドが高くなる。また、プロセッサ1 b では2 つのサーバ・プロセス群1 2 a ～1 2 d を実行しなければならないので、プロセッサ1 b が過負荷状態になる可能性がある。プロセッサ1 c あるいは1 d は、適切な負荷状態出運用されているので、プロセッサ1 b だけが過負荷になれば、負荷の不均等な状態が生じることになる。

【 0 0 1 3 】 本発明は、上記実情に鑑みてなされたものであり、プロセッサ障害等によるプロセッサ間でのプロセス再配置が行われた際に、効率的な負荷分散を行うことを可能とする負荷分散方式を提供することを目的とする。

【 0 0 1 4 】

【 課題を解決するための手段】 本発明は、複数のプロセッサ各々の制御下に、クライアント・プロセスと、このクライアント・プロセスから要求された手続きを所定のメモリ 領域を共有しながら実行する複数のプロセスからなるサーバ・プロセス群とを配置し、いずれかのプロセッサに障害が発生したときに、そのプロセッサ下に配置されていた上記クライアント・プロセス及びサーバ・プ

4

ロセス群を他の特定のプロセッサ下に再配置させることにより 処理を継続するマルチプロセッサ計算機において、上記クライアント・プロセス及びサーバ・プロセス群がいずれのプロセッサ下に配置されているのかを監視するプロセス監視手段と、上記クライアント・プロセスから要求された手続きを所定のサーバ・プロセス群に分配する手続き分配手段と、上記プロセス監視手段の監視結果により 複数のクライアント・プロセス及びサーバ・プロセス群が配置されているプロセッサを検出した場合に、この複数のクライアント・プロセスから要求された手続きが一つのサーバ・プロセス群に分配されるように上記手続き分配手段を制御する制御手段とを具備してなることを特徴とする。

【 0 0 1 5 】 また、本発明は、各プロセッサの負荷を測定する負荷測定手段と、この負荷測定手段の測定結果によりプロセッサ間に負荷の偏りがあると判断した場合に、負荷の高いプロセッサ下のサーバ・プロセス群に本来分配されるべき手続きの一部を負荷の低い他のプロセッサ下のサーバ・プロセス群へ分配されるように上記手続き分配手段を制御する制御手段とを具備してなることを特徴とする。

【 0 0 1 6 】

【 作用】 本発明によれば、図7 に示したように一つのプロセッサで複数のサーバ・プロセス群1 2 a ～1 2 b を動作させている場合に、クライアント・プロセス5 a ～5 b からの手続きを一つのサーバ・プロセス群、例えばサーバ・プロセス群1 2 b のキューのみへ引き渡すように分配制御する。これにより、一つのプロセッサ下では一つのサーバ・プロセス群だけが動作することになり、即ち、そのデータベース・バッファだけが使用されることになる。つまり、もう一方のデータベース・バッファは使用されていない状態なので、二つのバッファ間でのデータの一貫性を保証する必要はなくなり、その処理に必要なオーバーヘッドをなく することができる。

【 0 0 1 7 】 また、障害を発生させたプロセッサ1 a 下に配置されていたサーバ・プロセス群1 2 a の実行を引き継いだプロセッサ1 b には、それまでに実行していたサーバ・プロセス群1 2 b の負荷に加えて、新たに実行するサーバ・プロセス群1 2 a の負荷がかかる。したがって、上述のようにプロセッサ1 b 下で動作するサーバ・プロセス群をサーバ・プロセス群1 2 b の一つだけに限定したとしても、プロセッサ1 b が過負荷状態になることがある。

【 0 0 1 8 】 本発明によれば、図7 においてプロセッサ1 b が過負荷状態になった場合に、クライアント・プロセス5 a から要求された手続きの一部をプロセッサ1 c ～1 d のサーバ・プロセス群1 2 c ～1 2 d のキューへ割り 振るようにする。これによりプロセッサ1 b の負荷を軽減させると共に、プロセッサ1 c ～1 d の負荷を増大させることになり、システム全体として負荷の均等化

10

20

30

40

50

5

を実現することができる。

【 0 0 1 9 】

【 実施例 】 以下図面を参照して本発明の一実施例を説明する。図1 は同実施例に係るシステムの概略構成を示す図である。同システムでは、それぞれのサーバ・プロセス群1 2 a ~ 1 2 d が実行する手続きについて、他のすべてのサーバ・プロセス群1 2 a ~ 1 2 d が相互に実行できるようにするため、その手続きを内容は同じで名前が異なるような構成として登録しておく。ここでは、説明を簡単にするために、図1 における手続き8 a ~ 8 d を名前は異なるが内容が等しい手続きであるものとする。

【 0 0 2 0 】 図1 において、負荷測定部1 0 は、定期的に、あるいは手続き分配部6 からの指示により各プロセッサ1 a ~ 1 d の負荷を測定し、その結果を共有メモリ9 へ書き込む。プロセッサ1 a ~ 1 d の負荷としては、プロセッサ使用率やレディ・キューの長さ等を用いることが考えられる。ここではプロセッサ1 a ~ 1 d の使用率をオペレーティング・システムが提供するコマンドにより測定するものとする。

【 0 0 2 1 】 プロセス監視部1 1 は、サーバ・プロセス群1 2 a ~ 1 2 d が配置されているプロセッサがどれであるかを監視しており、一つのプロセッサで複数のサーバ・プロセス群を実行していることを見つけた場合、手続き分配部6 にその旨を伝える。また、プロセス監視部1 1 は、プロセッサ1 a ~ 1 d の障害等によりプロセッサ間でのプロセスの再配置が行われたときに起動される。

【 0 0 2 2 】 手続き分配部6 は、変換テーブル7 を保持し、この変換テーブル7 に記述された内容にしたがって手続きを変換し、その手続きを適切なサーバ・プロセス群のキューへ割り振る。また、手続き分配部6 は、プロセス監視部1 1 からの指示を受けて起動され、変換テーブル7 を変更する。そして、変換テーブル7 を変更した後に、負荷測定部1 0 に各プロセッサ1 a ~ 1 d の負荷を測定させる。その結果、プロセッサ1 a ~ 1 d の負荷に偏りがある場合には、変換テーブル7 を変更する。

【 0 0 2 3 】 なお、ここでは、手続き分配部6 をクライアント・プロセス5 a ~ 5 d とは別のプロセスとしているが、クライアント・プロセス5 a ~ 5 d で実行されるプログラムの一部に手続き分配部6 の動作を含めることも可能である。

【 0 0 2 4 】 図1 において、プロセッサが全て正常な状態のときは、サーバ・プロセス群1 2 a ~ 1 2 d はそれぞれプロセッサ1 a ~ 1 d に割り付けられて、クライアント・プロセス5 a ~ 5 d から要求された手続き8 a ~ 8 d をそれぞれ実行しているものとする。このときの手続き分配部6 の変換テーブル7 は、図2 に示す通りである。また、負荷測定部1 0 は、各プロセッサ1 a ~ 1 d の使用率を定期的に測定して、その結果を共有メモリ9

6

へ記録する。この値は、負荷の再分散を行うときに用いる。ここでは、各プロセッサ1 a ~ 1 d の使用率はすべて6 0 %程度であるとする。

【 0 0 2 5 】 いま、プロセッサ1 a に障害が発生し、フォールト・トレラント 計算機のリカバリ 機能によりクライアント・プロセス5 a とサーバ・プロセス群1 2 a とがプロセッサ1 b に再配置されたとする。プロセス監視部1 1 は、サーバ・プロセス群1 2 a ~ 1 2 d が配置されているプロセッサを調べ、プロセッサ1 b 下でサーバ・プロセス群1 2 a ~ 1 2 b の二つのサーバ・プロセス群が配置されていることを検出して、手続き分配部6 へプロセッサ1 b 下で二つのサーバ・プロセス群が実行されている旨を通知する。

【 0 0 2 6 】 プロセス監視部1 1 からの通知を受けた手続き分配部6 は、変換テーブル7 を図3 に示すように変更し、クライアント・プロセス5 a から要求された手続き8 a をすべて手続き8 b に変換してサーバ・プロセス群1 2 b のキューへ割り当てるようにする。これにより、サーバ・プロセス群1 2 a のデータベース・バッファ1 3 a は使われなくなり、データベース・バッファ1 3 a ~ 1 3 b 間でのデータの一貫性を保証する処理が必要なくなる。

【 0 0 2 7 】 ここで、プロセッサ1 b ~ 1 d の負荷がほぼ均一であれば問題ない。しかし、通常は全てのプロセッサが正常に動作している場合を想定して負荷分散がなされているので、プロセッサ1 a の負荷を引き継いだプロセッサ1 b の負荷が、他のプロセッサ1 c ~ 1 d の負荷よりも高くなる可能性が高いと考えられる。

【 0 0 2 8 】 そこで、手続き分配部6 は、負荷測定部1 0 へ各プロセッサ1 b ~ 1 d の使用率を測定することを指示する。負荷測定部1 0 の測定の結果、プロセッサ1 b の負荷が他のプロセッサ1 c ~ 1 d に比べて顕著に大きいことが判明した場合に、手続きの分配部6 が負荷の再分散を行う。例えば、プロセッサ1 b の使用率が1 0 0 %で、それ以外のプロセッサ1 c ~ 1 d の使用率が6 0 %程度であったとすると、プロセッサ1 b の負荷をプロセッサ1 c ~ 1 d へ分散させるように、手続き分配部6 の変換テーブル7 を変更する。例えば、正常時にプロセッサ1 b ~ 1 d の使用率がほぼ等しかったとすると、プロセッサ1 a にかかっていた負荷を三つのプロセッサ1 b ~ 1 d に均等に分散すればよいことになる。そこで、手続き分配部6 は、変換テーブル7 を図4 に示すように変更する。

【 0 0 2 9 】 ここで、プロセッサに障害が起こった場合の動作を説明する。図5 は、同実施例の動作を説明するためのフローチャートである。まず、各プロセッサ1 a ~ 1 d が実行しているサーバ・プロセス群の数を調べる(図5 のステップA 1)。サーバ・プロセス群1 2 a ~ 1 2 d を特定のプロセッサ1 a ~ 1 d で実行するため、オペレーティング・システムに何らかの機構が用意

10

20

30

40

50

されているはずである。そして、この機構を用いることによりサーバ・プロセス群1 2 a ~ 1 2 d がどのプロセッサで実行されているかを知ることができる。

【 0 0 3 0 】次に、複数のサーバ・プロセス群を実行しているプロセッサがあるかどうかを判定する(図5 のステップA 2)。通常、すべてのプロセッサ1 a ~ 1 d でサーバ・プロセス群1 2 a ~ 1 2 d のいずれかが実行されているので、あるプロセッサが故障して、そのプロセッサ下に配置されているサーバ・プロセス群が他のプロセッサへ再配置された場合には、そのプロセッサでは複数のサーバ・プロセス群を実行していることになる。このため、プロセッサ障害発生時には、ほとんどの場合この条件に該当することとなる(図5 のステップA 2 の Y)。

【 0 0 3 1 】複数のサーバ・プロセス群が実行されているプロセッサが存在する場合(図5 のステップA 2 の Y)、これまでそれぞれのサーバ・プロセス群へ割り当てていた手続きを一つのサーバ・プロセス群へ割り当てるように、手続き分配部6 の変換テーブル7 を変更する(図5 のステップA 3)。

【 0 0 3 2 】次に、各プロセッサ1 a ~ 1 d の使用率を調べ(図5 のステップA 4)、プロセッサの使用率に偏りがないか調べる(図5 のステップA 5)。故障プロセッサに対する負荷を引き継いだプロセッサの使用率は、他のプロセッサ使用率に比べて高くなるので、ほとんどの場合は該当することとなる(図5 のステップA 5 の Y)。

【 0 0 3 3 】ここで、故障が起こる前の各プロセッサ1 a ~ 1 d の使用率に基づいて、故障プロセッサの負荷をどのように他のプロセッサへ分配するかを決定する(図5 のステップA 6)。なお、本発明は、ここに示した実施例に限定されるものではなく、その請求範囲を逸脱しない範囲で適宜変更可能である。

【 0 0 3 4 】

【 発明の効果 】上述したように本発明によれば、フォールト・トレランスを実現するプロセッサ計算機においてトランザクション処理を行う場合に、プロセッサが障害を発生させたためにそのプロセッサで実行されていたサーバ・プロセス群を他のプロセッサに引き継がれた時の負荷分散を効率よく実現できる。

【 図2 】

変換前	変換後	割合
8a	8a	1 0 0 %
8b	8b	1 0 0 %
8c	8c	1 0 0 %
8d	8d	1 0 0 %

【 0 0 3 5 】サーバ・プロセス群の実行を引き継いだプロセッサ下で複数のサーバ・プロセス群を実行すると、複数のデータベース・バッファ間のデータの一貫性を保証するための処理が必要となり、そのプロセッサの負荷が高くなる。

【 0 0 3 6 】本発明では、複数のクライアント・プロセスからの手続きを一つのサーバ・プロセス群のキューのみに割り振ることにより、この問題を解決している。また、サーバ・プロセス群の実行を引き継いだプロセッサには、故障プロセッサにかかっていた負荷が追加されるため、他のプロセッサに比べて負荷が高くなる。

【 0 0 3 7 】本発明では、負荷の高いプロセッサのサーバ・プロセス群のキューへ割り振られていた手続きを他のプロセッサのサーバ・プロセス群のキューへ割り振られるように変更することにより、負荷の均等化を図ることができる。

【 図面の簡単な説明 】

【 図1 】同実施例に係るシステムの概略構成を示す図。

【 図2 】プロセッサが正常な時の変換テーブルの例を示す図。

【 図3 】障害プロセッサで実行していたサーバ・プロセス群を別のプロセッサで動作させた後の変換テーブルの例を示す図。

【 図4 】負荷を分散した後の変換テーブルの例を示す図。

【 図5 】同実施例の動作を説明するためのフローチャート。

【 図6 】従来のフォールト・トレラント 計算機でトランザクション・モニタを動作させた時のプロセス構成を示す図。

【 図7 】従来のフォールト・トレラント 計算機でプロセッサに障害が起こった時のプロセスの配置を示す図。

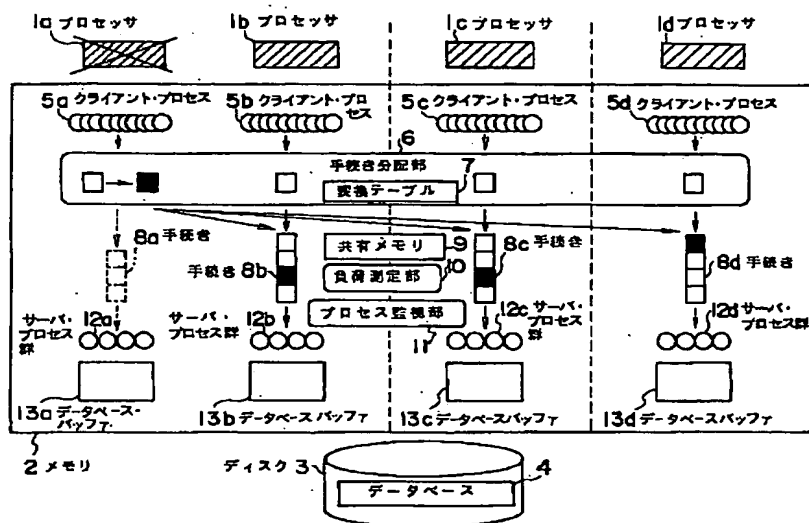
【 符号の説明 】

1 a ~ 1 d …プロセッサ、2 …メモリ、3 …ディスク、4 データベース、5 a ~ 5 d …クライアント・プロセス、6 …手続き分配部、7 …変換テーブル、8 a ~ 8 d …手続き、9 …共有メモリ、1 0 …負荷測定部、1 1 …プロセス監視部、1 2 a ~ 1 2 d …サーバ・プロセス群、1 3 a ~ 1 3 d …データベースバッファ。

【 図3 】

変換前	変換後	割合
8a	8b	1 0 0 %
8b	8b	1 0 0 %
8c	8c	1 0 0 %
8d	8d	1 0 0 %

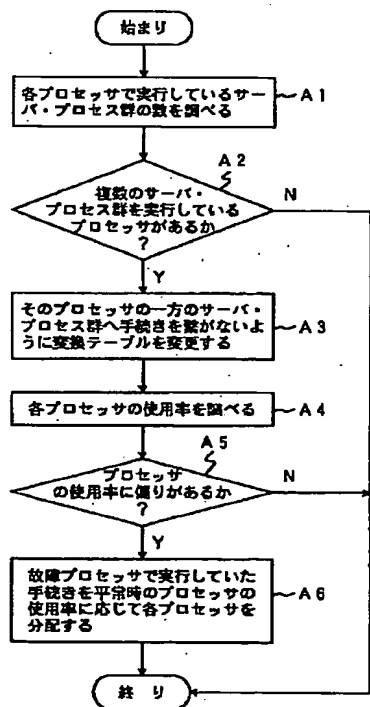
【 図1 】



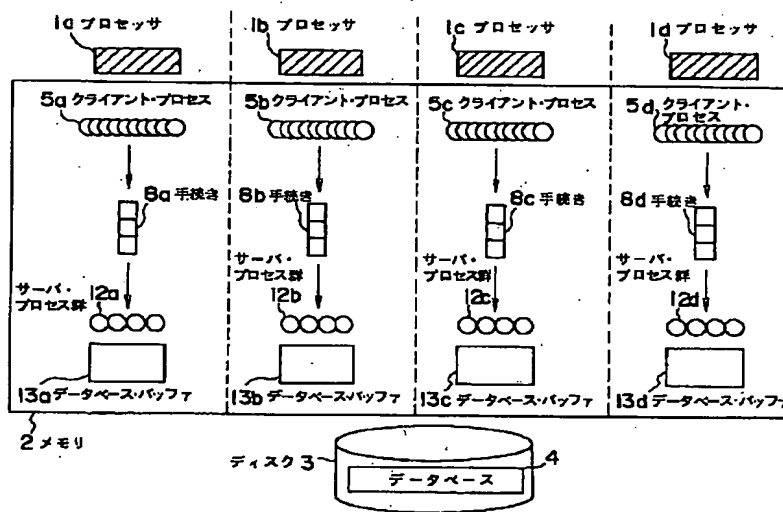
【 図4 】

変換前	変換後	割合
8a	8b	33%
	8c	33%
	8d	33%
8b	8b	100%
8c	8c	100%
8d	8d	100%

【 図5 】



【 図6 】



【 図7 】.

